CLAIM AMENDMENTS

Claim Amendment Summary

Claims pending

- Before this Amendment: Claims 1-41
- After this Amendment: Claims 1-38 and 41

Non-Elected, Canceled, or Withdrawn claims: 39-40

Amended claims: none

New claims: none

Claims:

1. (Original) In a host of a virtual machine environment having one or more methods in a shared managed library, a process comprising allowing or disallowing a call from a first managed code caller to a first said method based upon a configuration of the first said method with a hosting rule selected from the group consisting of:

always allow any said call from any said managed code caller to the first said method;

never allow any said call from any said managed code caller to the first said method due to the method's inappropriateness for the runtime environment; and

conditionally allow any said call from any said managed code caller to the first said method.

2. (Original) The process as defined in Claim 1, the conditional allowance is based upon:

the method's required level of trust; and

a level of trust attributed to the first managed code caller.

3. (Original) The process as defined in Claim 2, wherein the allowing or disallowing a call further comprises:

compiling code corresponding to the first managed code caller into native code; and

executing the native code corresponding to the first managed code caller during which the call from the first managed code caller to the first said method native code is made.

4. (Original) The method as defined in Claim 3, further comprising throwing an exception during the execution when the call from the first managed code caller to the first said method native code is made and:

the call that is never allowed; or

the level of trust attributed to the first managed code caller is insufficient when compared to a security permission demand assigned to and required by the first said method.

5. (Original) The process as defined in Claim 1, wherein when the call from the first managed code caller is allowed, access is provided by the first said method to a protected resource.

6. (Original) The process as defined in Claim 1, wherein any said allowed call provides any said managed code caller with access to one or more protected resources corresponding to the called said method.

7. (Original) The process as defined in Claim 1, wherein the level of trust attributed to the first managed code caller corresponds to an identity of a provider of the first managed code caller.

8. (Original) The process as defined in Claim 1, wherein the host compiles the first managed code caller into native code that is executed by a common language runtime via the host's operating system.

9. (Original) The process as defined in Claim 1, further comprising configuring each said method in the shared managed library with one said hosting rule.

10. (Original) The process as defined in Claim 9, wherein each said method receives the configuring prior to any said call to any said method from any said managed code caller.

11. (Original) The process as defined in Claim 1, further comprising:

determining whether the host will use any said hosting rule in allowing a call from any said managed code caller to any said method; and

when the determination is affirmative, configuring one or more said methods in the shared managed library with one said hosting rule.

12. (Original) The process as defined in Claim 11, wherein:

each said method in the shared managed library provides access to one or more protected resources; and

the host has access to a host configuration data structure comprising:

resource checking data for making the determination;

configuration data referencing the one or more protected resources and specifying:

each said protected resource to which access will always be allowed to any said managed code caller;

each said protected resource to which access will never be allowed to any said managed code caller; and

each said protected resource to which access will be allowed to any said managed code caller having a recognized level of trust satisfying a security permission demand corresponding to the protected resource;

the process further comprises:

accessing the host configuration data structure; and

using the resource checking data in the host configuration data structure to make the determination, wherein the configuring of one or more said methods in the shared managed library with one said hosting rule comprises, for each said method:

matching each said protected resource to which the method provides access to the corresponding protected resource in the host configuration data structure; and

for each said match, assigning to the method the corresponding configuration data that is associated with the protected resource in the host configuration data structure.

13. (Original) A computer readable medium including machine readable instructions for implementing the process as defined in claim 1.

14. (Original) A method comprising:

intercepting, with a host operating in a managed environment, a call from a managed caller to a managed callee; and

deriving whether the call is permissible according to the host's prior configuration of a plurality of said managed callees, wherein:

each said managed callee provides access to a protected resource; and

the prior configuration specifies whether to:

always allow the call to be made;

never allow the call to be made;

allow the call to be made based upon the degree to which the host trusts the managed caller.

15. (Original) The method as defined in Claim 14, wherein when the call is permissible, the managed callee provides access to the corresponding protected resource to the managed caller.

16. (Original) The method as defined in Claim 14, wherein the degree to which the host trusts the managed caller corresponds to an identity of a provider of the managed caller.

17. (Original) The method as defined in Claim 14, wherein the host compiles the managed caller into native code that is executed by a common language runtime via the host's operating system.

18. (Original) The method as defined in Claim 17, further comprising throwing an exception when, during the execution of the compiled native code corresponding to any said managed caller:

any said call that is never allowed to be made is attempted; or

any said call that is attempted when the degree to which the host trusts the managed caller is insufficient.

19. (Original) The method as defined in Claim 14, further comprising, prior to the intercepting:

determining whether the host will make the derivation; and

performing the intercepting and the deriving if the determination is affirmative.

20. (Original) The method as defined in Claim 19, wherein:

the host has access to a host configuration data structure comprising:

resource checking data for making the determination; and

configuration data sufficient for the host's prior configuration of the plurality of said managed callees;

the determining whether the host will make the derivation comprises accessing, with the host, the resource checking data in the host configuration data structure.

21. (Original) A computer readable medium including machine readable instructions for implementing the method as defined in claim 14.

22. (Original) An apparatus comprising:

virtual machine means, in a managed code portion including a plurality of method in a shared managed library, for operating a plurality of managed code callers in the managed code portion;

execution engine means, in a native code portion, for the virtual machine means;

means, in a native code portion, for providing an operating system; and

means for allowing or disallowing a call from a first said managed code caller to a first said method based upon a configuration of the first said method with a hosting rule selected from the group consisting of:

always allow any said call from any said managed code caller to the first said method;

never allow any said call from any said managed code caller to the first said method due to the method's inappropriateness for the runtime environment; and

conditionally allow any said call from any said managed code caller to the first said method based upon:

the configurations an assignment of the method's required level of trust; and

a level of trust attributed to the managed code caller.

23. (Original) The apparatus as defined in Claim 22, wherein the level of trust attributed to the managed code caller is based upon an identification (ID) of the provider of the managed code caller.

24. (Original) The apparatus as defined in Claim 22, further comprising:

means for compiling each said managed code caller from an intermediate language code and metadata into native code;

means for loading the native code with a Common Language Runtime (CLR) loader in the native code portion to load the compiled native code;

and

means for executing the compiled native code in the native code portion such that each said managed code caller can call one said method.

25. (Original) The apparatus as defined in Claim 22, further comprising means for throwing an exception when a disallowed call is attempted during the execution of the compiled native code corresponding to any said managed code caller.

26. (Original) The apparatus as defined in Claim 22, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata

generated by a language compiler, are represented by one or more of said managed code callers.

27. (Original) The apparatus as defined in Claim 22, wherein the execution engine means in the native code portion further comprises a compiler to compile each said managed code caller into native code for execution by the native code portion.

28. (Original) The apparatus as defined in Claim 22, wherein the execution engine means in the native code portion further comprises:

a Just In Time (JIT) compiler to compile each said managed code caller into native code; and

a CLR loader to load the compiled native code for execution by the native code portion.

29. (Original) A computing device comprising:

a managed code portion including:

one of more methods in a shared managed library;

one or more assemblies placed in respective application domains for execution; and

a virtual machine;

a native code portion including:

an execution engine for the virtual machine; and

an operating system under the execution engine;

logic configured to:

intercept a call from one said assembly to one said method; and

deriving whether the call is permissible according to a prior configuration of the one of more methods, wherein:

each said method provides access to a protected resource; and

the prior configuration specifies whether to:

always allow the call to be made;

never allow the call to be made;

allow the call to be made based upon the degree to which the one said assembly is trusted by the computing devices.

30. (Original) The computing device as defined in Claim 29, wherein when the call is permissible, the one said method provides to the one said assembly access to the corresponding protected resource.

31. (Original) The computing device as defined in Claim 29, wherein the degree to which the host trusts the managed caller corresponds to an identity of a provider of the one said assembly.

32. (Original) The computing device as defined in Claim 29, wherein the computing device compiles the one said assembly into native code that is executed by a common language runtime via the operating system.

33. (Original) The computing device as defined in Claim 32, further comprising throwing an exception when, during the execution of the compiled native code corresponding to the one said assembly:

the call that is never allowed to be made is attempted; or

the call that is attempted when the degree to which the computing device trusts the one said assembly is insufficient.

34. (Original) The computing device as defined in Claim 29, further comprising, prior to the intercepting:

determining whether the computing device will make the derivation; and

performing the intercepting and the deriving if the determination is affirmative.

35. (Original) The computing device as defined in Claim 34, wherein:

the computing device has access to a host configuration data structure comprising:

resource checking data for making the determination; and

configuration data sufficient for the computing device's prior configuration of the one of more methods;

the determining whether the computing device will make the derivation comprises accessing, with the computing device, the resource checking data in the host configuration data structure.

36. (Original) The computing device as defined in Claim 29, wherein the logic is further configured to receive intermediate language code and metadata generated by a language compiler to form the one or more assemblies for placement within respective application domains for execution.

37. (Original) The computing device as defined in Claim 36, wherein the intermediate language code and metadata generated by the language compiler from one or more files each having a file type and being associated with user code.

38. (Original) The computing device as defined in Claim 29, wherein the execution engine further comprises:

a JIT complier to compile said assemblies into native code; and

a CLR loader to load the compiled native code for execution in the native code portion.

39. (Withdrawn) A host comprising logic for a runtime environment using a host configuration data structure containing host configuration information to disallow a call to a managed code callee from an untrusted managed code caller or to disallow a call to a managed code callee that is deemed inappropriate for the runtime environment.

40. (Withdrawn) The host as defined in Claim 39, wherein:

the host compiles the managed code caller into native code that is executed by a common language runtime via the host's operating system; and

a host protection exception is thrown during the execution of the native code when the call to the managed code callee:

is made by the untrusted managed code caller; or

is deemed inappropriate for the runtime environment.

41. (Original) A host operating in a managed environment and comprising:

logic for configuring each of a plurality of managed callees, each providing access to a protected resource, with a configuration that:

always allow a call to be made to the managed callee for access to the corresponding protected resource;

never allow a call to be made to the managed callee for access to the corresponding protected resource; or

allow a call to be made to the managed callee for access to the corresponding protected resource based upon trust of the host for a managed caller;

logic for intercepting a call from a particular said managed caller to a particular said managed callee; and

logic, after intercepting the call, for determining whether the call is permissible according to the configuration of the particular managed callee